VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

POROVNÁNÍ VLASTNOSTÍ DISTRIBUOVANÝCH SOUBOROVÝCH SYSTÉMŮ

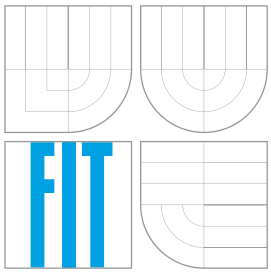
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN TOMEC

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

POROVNÁNÍ VLASTNOSTÍ DISTRIBUOVANÝCH SOUBOROVÝCH SYSTÉMŮ

COMPARISON OF DISTRIBUTED FILE SYSTEM FEATURES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN TOMEC
AUTHOR

VEDOUCÍ PRÁCE — Ing. IGOR SZÖKE
SUPERVISOR

BRNO 2008

Zadání bakalářské práce/3181/2007/xtomec05

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Tomec Martin**
Obor: Informační technologie
Téma: **Porovnání vlastností distribuovaných souborových systémů**
Kategorie: Počítačové sítě

Pokyny:

1. Vyberte si několik distribuovaných souborových systémů (DFS) a seznamte se s jejich vlastnostmi a možností nastavení.
2. Vybrané systémy zkompilujte a nainstalujte na testovacím HW.
3. Navrhněte vhodný soubor testovacích úloh.
4. Proveďte sadu testů na všech vybraných DFS. Zaměřte se zejména na rychlost operací čtení/zápis a odolnost proti výpadku.
5. Porovnejte výsledky testů a zhodnoťte vlastnosti zvolených DFS.

Literatura:

- Dle pokynu vedouciho

Při obhajobě semestrální části projektu je požadováno:

- Bod 1 a alespoň část bodů 2 a 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Szöke Igor, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Pavel Zemčík
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Martin Tomec**
Id studenta: 78706
Bytem: Mírová 924/4, 674 01 Třebíč
Narozen: 11. 04. 1986, Třebíč
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Porovnání vlastností distribuovaných souborových systémů
Vedoucí/školitel VŠKP: Szöke Igor, Ing.
Ústav: Ústav počítačové grafiky a multimédií
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Tato práce se zabývá porovnáním distribuovaných souborových systémů Ceph, Lustre a Kosmos FS. Jsou zde shrnuty obecné vlastnosti těchto systémů, včetně způsobu jejich instalace, konfigurace a zprovoznění. Dále jsou popsány testovací nástroje a sada testů, kterými jsem testoval výkon a možnosti těchto systémů na síti devíti počítačů. V poslední části jsem zpracoval a vyhodnotil výsledky těchto testů. Hodnocení bylo zaměřeno na určení omezujících míst systémů. Z těchto systémů se jako nejvhodnější jevil Lustre.

Klíčová slova

distribuovaný souborový systém, Lustre, Ceph, Kosmos

Abstract

This work deals with comparsion of distributed filesystems Ceph, Lustre and Kosmos FS. General properties of these systems are summarized here, including method of their installing, configuring and launching. Testing tools and test sets are described in the next part. Performance and abilities of these systems were tested on a network of nine computers. Last part contains analyses and evaluation of results, figuring out bottleneck of these systems. Lustre seems to be the best of these systems.

Keywords

distributed file systems, Lustre, Ceph, Kosmos

Citace

Martin Tomec: Porovnání vlastností distribuovaných souborových systémů, bakalářská práce, Brno, FIT VUT v Brně, 2008

Porovnání vlastností distribuovaných souborových systémů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Igora Szökeho. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Tomec
14. května 2008

Poděkování

Děkuji Vítězslavu Humpovi za spolupráci při testování, Ing. Tomáši Kašpárkovi za odbornou pomoc a zapůjčení hardware a Ing. Igoru Szökemu za konzultace a vedení.

Obsah

Kapitola 1

Úvod

I když se výkon dnešních osobních počítačů stále zvyšuje, je často výhodné zapojit je do jedné sítě a využít jejich společný výkon. Zejména pokud se jedná o pevné disky, může být spojení do sítě výhodné, protože rychlosti pevných disků jsou obecně nižší než rychlosti sítě. Aby však bylo možné přistupovat přes síť k těmto diskům, je potřeba použít síťový souborový systém. Klasické síťové systémy (jako například NFS) však striktně dodržují architekturu klient – server, takže je vždy využíván disk pouze na jednom serveru.

Distribuované souborové systémy jsou, na rozdíl od klasických síťových, rozložené mezi několik serverů. V závislosti na implementaci je toto rozložení výhodné z důvodu spolehlivosti, výkonu nebo dostupnosti (například u geograficky rozsáhlých systémů). Samostatnou větví jsou pak paralelní distribuované souborové systémy, které se zaměřují na maximální výkon při současném přístupu více klientů. Často také data replikují na více serverů, čímž je kromě vyšší rychlosti čtení (na úkor rychlosti zápisu) zajištěna i větší spolehlivost. Obecně jsou kladeny vyšší požadavky na rychlost čtení, jelikož v reálném nasazení objem čtených dat výrazně převyšuje objem zapisovaných dat.

Zaměřil jsem se na srovnání distribuovaných souborových systémů, které jsou volně dostupné. Zároveň se mnou zpracovával stejné téma kolega Vítězslav Humpa, takže jsme si tyto systémy rozdělili, abychom se jimi mohli zabývat podrobněji. V této práci jsem zpracoval systémy Lustre, Kosmos FS, Starfish a Ceph. Systémy PVFS2, GlusterFS, Gfarm Grid, dCache a MogileFS zpracoval ve své bakalářské práci [4] kolega.

Vybrané souborové systémy jsem zhodnotil nejprve teoreticky podle dostupné dokumentace. Získané informace jsou sepsány v první kapitole. Dále jsme s kolegou tyto souborové systémy nainstalovali na síti devíti počítačů a testovali jejich vlastnosti v praxi. Popis těchto testů a jejich výsledky jsou shrnuty v dalších dvou kapitolách.

Kvůli plánovanému nasazení distribuovaného souborového systému na naší fakultě, jsme testovali na operačním systému CentOS, abychom zároveň zjistili, jak budou systémy použitelné v prostředí fakulty.

Kapitola 2

Obecný popis

Následující kapitola popisuje obecné vlastnosti testovaných souborových systémů. Jde o shrnutí veřejně dostupných informací, rozhodujících pro nasazení systému v praxi. Zajímá nás především, zda podporují přístupová práva, jak jsou připojitelné do souborového systému (někdy je potřeba knihovna funkcí pro přístup) a také kdo se podílí na vývoji (aby např. nešlo o „mrtvý projekt“). Zároveň je v této kapitole shrnutý postup instalace a konfigurace a to včetně problémů, na které jsem narazil. Je nutné podotknout, že většina těchto systémů je stále ve vývoji a tyto nedostatky jistě budou odstraněny, nicméně považuji za vhodné se o těchto chybách zmínit. Systémy jsou řazeny abecedně.

2.1 Ceph

Tento systém je vyvíjen na University of California, Santa Cruz. Jeho cílem bylo vyvinout paralelní distribuovaný souborový systém dostupný pod GPL a zaplnit tak mezeru na tomto poli (která v současné době již není tak velká). Je vyvíjen od roku 2005, ale stále není uvolněna stabilní verze vhodná pro produkční nasazení. Z vybraných systémů má Ceph asi nejobtížnější instalaci (resp. konfiguraci). Protože jde o výzkumný projekt, dokumentace [8] se zaměřuje na popis použitých algoritmů a principů a postup instalace a konfigurace je zdokumentován pouze formou vzorových příkladů na wikipedii. Neexistuje také zatím žádná oficiálně vydaná verze, lze pouze stáhnout některou z posledních revizí z git.

Systém vyhovuje standardu POSIX a prošel i většinou testů na shodu (vydanými 2.dubna 2008, blíže viz [7]). Striktní dodržování standardu však může v některých případech omezovat výkon systému. Například synchronní zápis několika klientů do jednoho souboru vyžaduje zamykání tohoto souboru, takže v jednu chvíli může zapisovat pouze jeden klient. Pokud však na aplikační úrovni zajistíme, aby klienti zapisovali do různých částí souboru, je toto zamykání zbytečné. Proto Ceph umožňuje globálně tyto zámky vypnout.

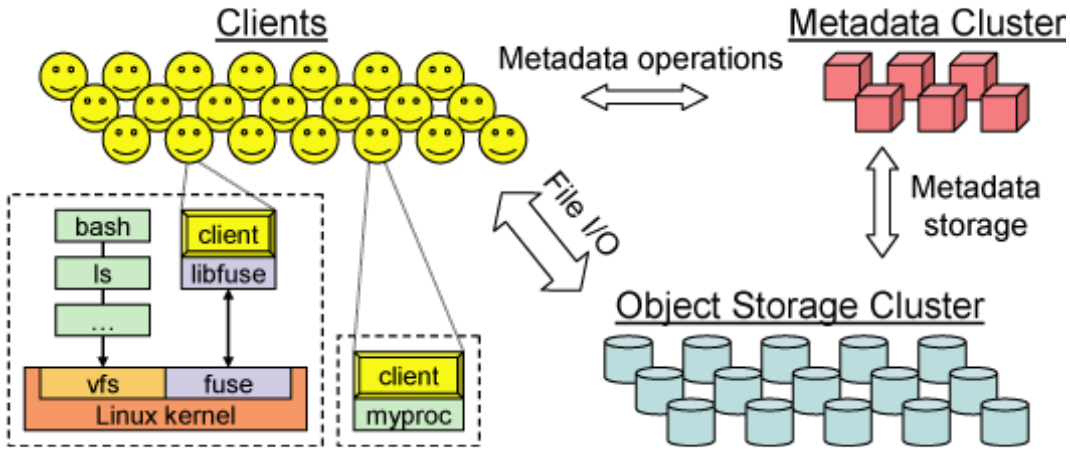
2.1.1 Architektura

Architektura systému nerozlišuje metadata od samotných dat. Obě skupiny dat se ukládají na jeden cluster složený většinou z více strojů. Tento cluster je autonomním systémem se samostatným řízením replikace a automatickým vyvažováním zátěže (systém RADOS [9]). Zátěž se vyvažuje průběžně a cluster je tedy schopný postupně se přizpůsobit aktuálnímu využití. Pro zvýšení výkonu se tedy využívá výpočetní kapacity stanic, což může být výhoda, pokud tyto stanice stojí jako samostatný úložný systém, nebo naopak nevýhoda, pokud jsou stanice zároveň součástí výpočetního systému. Redundance je zatím zajištěna

pouze replikací souborů, i když bylo původně v plánu implementovat zálohování pomocí parity.

Ceph se zaměřuje na práci s metadaty souborů. Podle dokumentace tvoří práce s metadaty polovinu zátěže typického souborového systému[1]. Na zpracování metadat není vyhrazen pouze jeden stroj, ale celý cluster. Tento cluster zpracovává požadavky na metadata a udržuje v paměti RAM cache posledních požadavků. Vlastní uložení a načítání metadat ponechává na datovém clusteru. Odpovědnost za metadata je rozdělována hierarchicky mezi jednotlivé uzly.

Komunikace s klienty i komunikace mezi stanicemi je postavena na protokolu TCP. Klient přistupuje k systému buď přes FUSE, nebo přes knihovnu funkcí, ale do budoucna je cílem zabudovat podporu přímo do jádra. Graficky je architektura znázorněna na obrázku 2.1 (obrázek byl převzat z [8]).



Obrázek 2.1: Architektura systému Ceph

Na discích serverů používá vlastní souborový systém, takže je vhodné mu vyhradit zvláštní diskový oddíl. Je sice možné místo diskového oddílu použít běžný soubor, ale tento způsob je určen spíše pro ladění, než pro běžný provoz. Kvůli rychlejšímu přístupu je možné (a doporučované) pro žurnál tohoto souborového systému použít jiný pevný disk, nebo NVRAM.

2.1.2 Kompilace

Pro kompilaci je nutný autoconf verze 2.59 a také knihovna openmpi ve verzi 1.2.5. Pomocí skriptu `autogen.sh` se vygenerují konfigurační skripty a poté již stačí „klasická“ posloupnost příkazů:

```
./configure
make
make install
```

[1] Tento závěr však vychází z [6], kde byla testována zátěž zejména při používání lokálních souborových systémů na pracovních stanicích

2.1.3 Konfigurace a spouštění

Popis konfigurace vychází pouze ze vzorových příkladů z wikipedie [1] a nemusí tedy být korektní a kompletní. Základem konfigurace je globální konfigurační soubor, u kterého Ceph vyžaduje, aby byl dostupný na všech serverech clusteru (například přes NFS). Tento soubor vytvoříme pomocí příkazů:

```
monmaptool --create --clobber --add 10.0.0.1:12345 --print .ceph_monmap
mkmonfs --clobber mondata/mon0 --mon 0 --monmap .ceph_monmap
```

kde 10.0.0.1 je IP adresa stroje na němž bude běžet monitorovací proces a mondata/mon0 je cesta k adresáři kam si bude ukládat informace o struktuře a stavu systému. Všechny následující příkazy musí být spouštěny v adresáři s tímto vygenerovaným konfiguračním souborem (nebo k němu mít přístup a nastavenou cestu). Samotný monitorovací proces pak spustíme příkazem:

```
cmon mondata/mon0 -o out -d --debug_mon 10 --debug_ms 1
```

Pomocí nástroje osdmaptool se vytváří konfigurační soubor pro servery datového clusteru[2]. Monitorovacímu procesu se pak pomocí cmonctl tento soubor nastaví jako mapa datových serverů:

```
osdmaptool --clobber --createsimple .ceph_monmap 4 --print .ceph_osdmap
cmonctl osd setmap -i .ceph_osdmap
```

Pak se na jednotlivých datových serverech naformátují diskové oddíly určené pro data:

```
cosd --mkfs_for_osd <num> /dev/sdb
```

kde `num` je pořadové číslo datového serveru (počínaje 0). Pokud chceme pro uložení žurnálu použít jiný disk, musí na něj existovat odkaz `/dev/sdb.journal`. Poté můžeme na jednotlivých strojích spustit procesy datových serverů příkazem cosd:

```
cosd /dev/sdb -o out -d --debug_osd 10
```

Složka out musí existovat, do ní se pak ukládají ladící výpisy. Nakonec spustíme proces metadata serveru:

```
cmds --debug_ms 1 --debug_mds 10 -o out -d
```

Tím je spuštěn celý systém, zbývá připojit klienty. K tomu slouží příkaz cfuse, který musí být spuštěn v adresáři s konfiguračním souborem (.ceph_monmap)

```
cfuse /mnt/ceph
```

2.2 Kosmos FS

Na základě uvolněných informací o proprietárním Google FS vytváří Sriram Rao se skupinou vývojářů podobný souborový systém. Systém je psán v jazyce C++ a je zatím ve vývoji, takže není doporučováno jeho produkční nasazení. Je vydán pod licencí „Apache 2.0 license“[3] , která umožňuje upravovat a šířit zdrojový kód. V době psaní práce je k dispozici

[2] Servery se označují jako Object-based Storage Device, z toho je tedy zkratka OSD
[3] Dostupná na http://www.apache.org/licenses/LICENSE-2.0.html

verze 0.1.2, což jistě vypovídá o tom, že nejde o stabilní a odladěnou verzi. Nicméně již tuto verzi je možné spustit, a bude vhodné zjistit, kde má slabá místa a porovnat je s ostatními systémy Nejde tedy o srovnání „stability“ ale spíše o porovnání možných řešení ukládání dat.

Podobně jako u Google FS se soubory dělí do takzvaných „chunks“, což jsou bloky o velikosti 64 MB. Je tedy zřejmé, že jde o systém optimalizovaný pro velké soubory. To odpovídá filozofii Google FS, která také předpokládá práci s několika málo velkými soubory. Protože je umožněno, aby několik klientů zapisovalo zároveň na konec jednoho souboru, je možné takto vytvářet rozsáhlé soubory. Není sice zajištěno, že pořadí zápisů zůstane zachováno, ale je zajištěna atomičnost jednotlivých zápisů. Tyto soubory jsou pak většinou čteny sekvenčně velkým množstvím klientů. Rozdělení na velké bloky je tedy pro tento účel ideální, protože výrazně omezuje režii. Pro velké množství malých souborů je však systém přímo nevhodný, jelikož by se příliš zvýšila zátěž metadata serveru. Další (samozřejmou) výhodou tohoto členění je prakticky libovolná velikost souboru (není omezená kapacitou jednotlivých serverů, ale celkovou kapacitou systému).

U každého uloženého bloku se ukládá i kontrolní součet, čímž se systém brání poruchám disku (příp. ovladače disku), které nejsou tak vzácné, jak by se mohlo zdát (viz [3]). Bloky jsou také několikrát replikovány (implicitně třikrát) pro případ výpadku celého stroje a pro zvýšení rychlosti čtení.

2.2.1 Architektura

Systém sestává ze dvou základních částí, metadata serveru a chunk serveru, které se spouští jako samostatné procesy. Na jednom stroji je tedy možné spustit obě části zároveň (případně i více instancí, pokud to má opodstatnění). Metadata server ukládá minimum informací, zejména názvy souborů a rozložení chunks mezi jednotlivé servery. Cílem je, aby se tato data vešla do paměti RAM a byla tedy dostupná bez zpoždění. Klienti od metadata serveru zjistí do kterých chunks je soubor rozložen a zároveň i rozmístění těchto chunks mezi servery. V současné verzi není metadata server replikován a zůstává tedy slabým místem z hlediska spolehlivosti systému, ale v dalších verzích je plánována náprava. Do té doby je možné metadata server zálohovat ručně. Chunk servery slouží k ukládání surových dat. Jednotlivé chunks ukládají jako běžné soubory do souborového systému, což může být v případě rozsáhlejších systémů omezující. Už v současné verzi byla přidána alespoň možnost rozdělit tyto soubory do podadresářů. Tyto servery mohou samozřejmě běžet na stejném stroji jako metadata server.

Servery mezi sebou a s klienty komunikují výhradně pomocí protokolu tcp, na straně klienta je zatím k systému přistupováno přes knihovnu funkcí. Podpora pro FUSE je sice plánována, ale v aktuální verzi je zatím nepoužitelná (příliš pomalá a implementuje pouze základní operace).

Systém se konfiguruje přes konfigurační soubor, podle nějž pak instalační skript rozmístí binární soubory a startovací skript spustí služby na jednotlivých serverech. Tento způsob má výhodu zejména v tom, že je možné celý systém spravovat z jednoho místa. Vývoj je zaměřen především na operační systém Fedora Core 5, pro nějž jsou distribuovány i binární soubory.

2.2.2 Kompilace

Požadované balíky jsou buď přímo součástí balíčkovacího systému, nebo jsou dostupné na http://rpm.pbone.net/

- gmake 3.81
- boost 1.34
- cmake 2.4.7
- log4cpp 0.3.5rc1

Pokud kompilujeme na 32-bitovém systému, jsou potřeba drobné úpravy v kódu. V další uvolněné verzi by však tyto chyby měly být již opravené. V souboru CMakeLists.txt je potřeba nastavit cesty `JAVA_INCLUDE_PATH` a `JAVA_INCLUDE_PATH2`.[4] Implicitně je generována ladící verze. Pro verzi bez ladících výpisů je potřeba tamtéž nastavit:

```
set (CMAKE_BUILD_TYPE Release)
```

V adresáři se zdrojovými soubory pak vytvoříme podadresář pro výsledné binární soubory:

```
mkdir build
cd build
```

Pomocí `cmake` nakonfigurujeme překlad a vygenerujeme makefile. Pak pomocí `gmake` přeložíme zdrojové soubory a pomocí `gmake install` dokončíme instalaci (mimo jiné zkopíruje binární soubory do podadresáře build/bin).

```
cmake -D CMAKE_BUILD_TYPE="Release" ../
gmake
gmake install
```

Na ostatní stanice se binární soubory zkopírují automaticky během konfigurace, stačí na ně tedy nainstalovat potřebné knihovny.

2.2.3 Konfigurace

Základem konfigurace je jeden soubor, ve kterém je popsána struktura celého systému. Popis struktury se zatím omezuje pouze na určení adres jednotlivých serverů a nastavení portu, na kterém mají naslouchat. Podle tohoto souboru se pomocí konfiguračních skriptů na jednotlivých stanicích spustí procesy serverů. Tyto skripty využívají ssh a je proto vhodné mít na všechny stanice zajištěn přístup bez hesla. Upravíme tedy vzorové konfigurační soubory ve složce `conf` a vytvoříme nový konfigurační soubor[5], např. `deploy.cfg`. Poté spustíme instalační skript (je nutné jej spouštět z adresáře scripts), který rozmístí potřebné binární soubory na všechny stanice:

```
python kfssetup.py -f ../conf/deploy.cfg -b ../build/bin
```

2.2.4 Spuštění

Pro spuštění (resp. zastavení) systému slouží skript kfslaunch.py. Ten podle konfiguračního souboru spustí (zastaví) procesy na jednotlivých stanicích. Stačí mu tedy zadat cestu ke konfiguračnímu souboru a parametr `--start` (`--stop`):

```
python kfslaunch.py -f ../conf/deploy.cfg --start
python kfslaunch.py -f ../conf/deploy.cfg --stop
```

[4] Pokud nepotřebujeme java rozhraní stačí prázdný řetězec.

[5] Vzorový konfigurační soubor je dostatečně komentovaný a patrně se ještě bude měnit, proto zde nemá smysl rozebírat jeho formát.

2.3 Lustre

Systém byl původně vyvíjený firmou Cluster File Systems, Inc., kterou v roce 2007 převzali Sun Microsystems. Díky tomu je kromě Linuxu plánována podpora i pro systém Solaris. Stále je vyvíjen pod licencí GNU GPL. Lustre je určen především pro rozsáhlejší clustery, což dokazuje například jeho použití na Blue Gene (viz [2]) a několika dalších významných superpočítačích. Primárně je určen pro stanice, které mají pole disků s redundancí – implementace nezajišťuje žádnou kontrolu chyb ani redundanci. Odolnost proti výpadku hardware je zajišťována až na úrovni jednotlivých serverů (viz dále).

Lustre je podporován především systémy Red Hat Enterprise Linux a SUSE. Pro tyto systémy jsou připravené předkompilované balíčky. Kvůli optimalizacím výkonu však vyžaduje úpravy v linuxovém jádře, což může být jeho nevýhodou. Pro aktuální verzi jádra existují hotové balíčky s již upraveným jádrem a stačí tedy nainstalovat balíček a upravit zavaděč. V případě, že z nějakých důvodů potřebujeme vlastní jádro, je možné aplikovat potřebné úpravy a překompilovat jádro s nimi. Modul pro klienta lze zkompilovat i pro neupravený kernel. Ztratí se tím sice část výkonu, ale odpadá nutnost u klienta upravovat kernel, což má samozřejmě výhody z hlediska údržby.

2.3.1 Architektura

Architektura systému je v principu podobná Google FS. Jeden metadata server (MDS[6]), na němž jsou uložena jména souborů, a jejich umístění. Dále několik úložných serverů (OSS), na nichž jsou uložena surová data. Na jednom fyzickém stroji obvykle (v závislosti na hardware) běží několik procesů úložných serverů (procesy se označují OST).

Metadata server má typicky svou záložní kopii, která jej nahradí v případě výpadku, a stejně tak datové servery mají možnost pracovat po dvojicích, kdy jeden server slouží zároveň jako záloha druhého. Implicitně však servery pracují samostatně, takže při výpadku některého serveru je část systému nedostupná. Zbytek systému však zůstává dostupný – výpadek se dotkne pouze některých souborů (nebo jejich částí, v případě využití prokládání). Systém rovněž detekuje výpadky klientů, aby uvolnil zámky nad jejich soubory a umožnil tak přístup ostatním klientům.

Komunikaci mohou kromě ethernetu zajišťovat i rozhraní Quadrics Elan, InfiniBand nebo Myrinet GM, díky abstraktní vrstvě LNET[7]. Zároveň se tím sníží vytížení procesoru na minimum, jelikož tato rozhraní podporují RDMA[8].

Od verze 1.6 se zásadně změnil způsob konfigurace systému a do výše uvedené architektury přibyl „configuration management server“ (MGS), což je proces spravující konfiguraci několika souborových systémů (stačí tedy většinou jediný, i v případě více nezávislých systémů). Mimo to se pro uložení dat místo ext3 začal používat vlastní souborový systém. Pro provoz každého serveru je tedy nutné vyhradit na disku zvláštní oddíl.

Každý soubor si vyhradí jeden inode[9] na metadata serveru a jeden inode na datovém serveru. Velikost oddílu pro metadata server je tedy nutné zvolit s ohledem na předpokládaný počet souborů (doporučením jsou 1-2% kapacity clusteru).

Na klientovi je díky úpravě jádra možné systém připojit transparentně příkazem mount. Pro vyšší výkon je poskytována knihovna pro přímý přístup a při použití této knihovny

[6] Zkratky používané v oficiální dokumentaci
[7] Lustre Networking, blíže viz [5]
[8] Remote Direct Memory Access, blíže opět v [5]
[9] Implicitně jde o úsek 4kB.

samozřejmě také odpadá nutnost u klienta upravovat kernel, nicméně je pak nutné upravovat klientskou aplikaci.

2.3.2 Instalace

Instalace je poměrně jednoduchá, veškeré potřebné balíky je možné stáhnout po bezplatné registraci na http://www.sun.com/software/products/lustre/get.jsp. Jednotlivé balíky:

- kernel-smp-<verze>.rpm - Balíček s upraveným a zkompilovaným jádrem. Je samozřejmě potřeba použít odpovídající verze modulů a utilit.
- kernel-source-<verze>.rpm – Balíček se zdrojovými soubory předchozího jádra. Při použití předkompilovaného jádra není potřeba.
- lustre-modules-<verze>.rpm – Potřebné moduly jádra.
- lustre-<verze>.rpm – Uživatelské nástroje pro konfiguraci systému. (a některé podpůrné nástroje)
- lustre-source-<verze>.rpm – Zdrojové soubory, včetně patche kernelu. Potřebné pouze pro kompilaci vlastních modulů, nebo vlastního jádra.
- lustre-ldiskfs-<verze> – Modul jádra pro vlastní souborový systém.
- e2fsprogs-<verze>.rpm – Běžné utility pro souborový systém. Je vyžadována verze 1.38 a vyšší.

Kromě těchto balíků je vyžadován perl (v jakékoliv „moderní“ verzi) a pro případnou kompilaci gcc (od verze 3.0). Kvůli závislostem je vhodné balíky instalovat v následujícím pořadí:

- e2fsprogs
- kernel-lustre-smp
- lustre
- lustre-ldiskfs
- lustre-modules

2.3.3 Konfigurace

Konfigurace byla od verze 1.6 značně zjednodušena. K základní konfiguraci slouží nástroj mkfs.lustre. Tímto nástrojem se vytvoří na zvoleném diskovém oddílu speciální souborový systém, který kromě vlastních dat obsahuje i informace o konfiguraci clusteru. Tento souborový systém pak stačí připojit na libovolné místo, čímž se spustí server.

Na jednom diskovém oddílu může být metadata server (MDS) zároveň s konfiguračním serverem (MGS). Souborový systém pro takovou kombinaci vytvoříme příkazem:

```
mkfs.lustre --fsname=testfs --mdt --mgs /dev/sdb1
```

Datový server (OSS) však vyžaduje svůj vlastní oddíl, který naformátujeme obdobným způsobem:

```
mkfs.lustre --fsname=testfs --ost --mgsnode=server@tcp0 /dev/sdb2
```

Parametr `--fsname` slouží k jednoznačnému přiřazení serveru ke konkrétnímu souborovému systému (na clusteru s jedním MGS může běžet více systémů) a parametr `--mgsnode` určuje adresu MGS a rozhraní pro připojení.

Za běhu lze systém konfigurovat pomocí nástroje `lctl`. Implicitně je například vypnuté prokládání souborů, což lze změnit příkazem:

```
lctl conf_param testfs-MDT0000.lov.stripecount=3
```

2.3.4 Spuštění

Servery se spouští připojením oddílu s jejich souborovým systémem. Lze je tedy spouštět automaticky při startu, pokud je přidáme do fstab, nebo ručně:

```
mount -t lustre /dev/sdaX /mnt/testX
```

Klient se z hlediska uživatele připojuje podobně jako jiné souborové systémy (pokud máme upravené jádro):

```
mount -t lustre server@tcp0:/testfs /mnt/testfs
```

2.4 Starfish

Souborový systém firmy Digital Bazaar, který firma začala vyvíjet jako náhradu za Lustre. Oproti Lustre systém Starfish nepožaduje úpravu jádra, beží celý v uživatelském režimu a je napsán v Pythonu. Jeho hlavní výhodou je úplná decentralizace (viz níže). V současné době je ve vývoji a není doporučováno jeho produkční nasazení (byť na tomto systému firma provozuje sklad multimédií). Jako stabilnější varianta je doporučován Glustre, jako rychlejší varianta Lustre [10]. Bohužel není vyvíjen pod GPL a pod současnou licencí je jeho použití omezeno na 1 TB uložených dat. Kvůli tomuto omezení a nestabilitě byl systém nakonec vyřazen z testování a je zde uveden pouze jako alternativa k předchozím.

2.4.1 Architektura

Architektura se zásadně liší od předchozích systémů. Neexistuje zde žádný centrální server, všechny uzly mezi sebou komunikují pomocí multicast vysílání a metadata jsou sdílena všemi uzly. Díky tomu je zajištěna maximální dostupnost – v případě výpadku jakéhokoliv počtu serverů zůstane zbylá část souborového systému dostupná. Zároveň tato architektura zajišťuje také snadné rozšíření systému – na novém stroji stačí spustit server a ten se sám připojí k dosavadní síti. Na druhou stranu takto decentralizovaný systém je obtížné ladit a omezenou stabilitu považuji u Starfish za největší nevýhodu.

2.4.2 Podpora

Starfish je primárně vyvíjen pro Debian, ale díky použití pythonu by měla být zajištěna maximální přenositelnost. V požadavcích má uvedeny vždy nejaktuálnější verze programů, ale systém fungoval i se staršími verzemi. Například s pythonem verze 2.4.3 byl systém dokonce stabilnější, než s verzí 2.5.1.

[10] Jak bylo uvedeno v diskuzi na http://lists.digitalbazaar.com:9081/pipermail/starfish-discussion/2007-October/000031.html

Kapitola 3

Postup testování

Systémy jsme testovali s kolegou Humpou na zapůjčeném hardware. Z porovnávaných systémů jsme si každý vybrali 3 nejperspektivnější, na které jsme se v testování zaměřili. Z vybraných systémů jsem tedy při testování vynechal Starfish, vzhledem k jeho nestabilitě a omezené licenci.

3.1 Testovací sestava

Testovací sestava byla složená z 9 počítačů, které měly všechny konfiguraci:

Intel Core 2 Duo 2.6GHz

2GB RAM

250GB HDD

1Gbit ethernet

Tyto stroje byly připojeny 1 Gbit ethernetem do jednoho společného přepínače. Na všech stanicích byl nainstalován operační systém CentOS 5, přičemž 8 strojů mělo jednotnou instalaci a poslední byl vyčleněn jako server.

Tento server poskytoval celé síti konfiguraci přes dhcp a sloužil obecně jako zázemí pro testovací skripty. Zároveň na něm byl vytvořen adresář dostupný ze všech stanic přes NFS. Tento adresář sloužil například pro umístění instalačních souborů (což usnadňovalo automatickou instalaci na všechny stanice) a zároveň byl využíván na shromažďování výsledků testů a statistik z jednotlivých stanic.

Disk jednotlivých stanic byl rozdělen na 3 oddíly. Pro systémový oddíl jsme vyhradili 50 GB a zbytek disku byl rozdělen mezi 2 datové oddíly. Zvolili jsme raději 2 oddíly, jelikož některé systémy vyžadují celý oddíl naformátovaný vlastním souborovým systémem. Protože jsme se na testovací sestavě střídali s kolegou, používal jsem výhradně třetí oddíl (tedy poslední). To může mírně snížit výkon, jelikož tento oddíl je nejblíže středu disku. Je tedy vhodné toto zohlednit v případě porovnání systémů s ostatními zpracovanými v práci kolegy[4].

3.1.1 Záznam statistik

Během jednotlivých testů bylo nutné zaznamenávat statistiky využití sítě, paměti, disků a procesoru pro pozdější analýzu. Na zaznamenávání těchto statistik jsme použili nástroj

HotSaNIC [1]. Tento nástroj zaznamenává (nejen) výše uvedené statistiky pomocí perlových skriptů a ukládá je ve formátu rrd. Nevýhodou tohoto nástroje je, že celá implementace počítá s pevně danou vzorkovací periodou 10 s. Ta je samozřejmě na některé testy příliš dlouhá, takže ji bylo nutné upravit na několika místech ve zdrojových souborech. Otázkou je zda častější spouštění skriptů neovlivní výsledky testů, ale i když jsme zvolili vzorkování jednou za sekundu, skripty nevyužili procesor na více než 2%. Jediné, čím mohly ovlivnit výsledky testů, byly přístupy na disk. Navíc by se tato chyba objevila u všech testovaných systémů a neovlivnila by tedy jejich srovnání.

Statistiky byly průběžně zaznamenávány na lokální disk a po skončení testu zkopírovány na server. Formát rrd totiž uchovává (v nejvyšším rozlišení) pouze několik posledních záznamů. Starší záznamy sumarizuje, aby soubor se záznamy zůstal rozumně velký. Výchozí nastavení nástroje HotSaNIC při zaznamenávání po jedné sekundě uchovává posledních 12 minut s rozlišením 1 s, což je dostatečné pro většinu testů. Drobným zklamáním u tohoto formátu pro mě byla jeho platformová závislost. Z údajů zaznamenaných na jednom systému nelze například generovat grafy na druhém systému.

Ze všech statistik, které nástroj HotSaNIC umožňuje zaznamenávat, byly vybrány pouze následující:

- Procesor (jednotlivá jádra i celkové vytížení)
- Paměť RAM
- Síťové rozhraní
- Rychlost čtení z disku a zápisu na disk (včetně počtu přístupů)

3.1.2 Testovací skripty

Testovací skripty z větší části vytvářel kolega Vítězslav Humpa, takže bych zde raději použil jeho popis skriptů (převzato z [4]):

Před uskutečněním testů jsme stáli před otázkou, jakým způsobem je nejlépe provést. Bylo potřeba vzít v potaz několik faktorů.

- Velikost clusteru, tedy počet aktuálně zapojených prvků DFS
- Synchronizaci testů v situacích, kdy museli připojení klienti provádět vybranou I/O operaci ve stejné chvíli
- Způsob zadání a vlastního spuštění testů

Za tímto účelem jsme napsali dva programy komunikující spolu pomocí tcp/ip a BSD sockets klasickou metodou klient/server. Jako implementační jakyk jsme zvolili Python. Ten jsme vybrali, protože je interpretovaný – nebylo nutné kompilovat při každé, během testování časté, změně kódu. Dále se jedná o jazyk vysokoúrovňový, dostatečně jednoduchý a přitom nabízející dobrou nízkoúrovňovou práci s BSD sockets a I/O operacemi.

První skript pojmenovaný „dfsSingleton“ jsme spouštěli jako klienta pouze na serveru. Spouštěl se zvlášť pro každý prováděný test. V tomto skriptu bylo nastavováno, jaké počítače hráli jakou roli podle momentálně aktivní konfigurace testovaného DFS. Dále zde

[1] Jako alternativní nástroj jsme zvažovali použití MRTG. Ten již také podporuje zaznamenávání do formátu rrd, ale v době testování nepodporoval nastavení libovolné periody záznamu. V aktuální verzi je podle dokumentace už možné nastavit periodu záznamu i v sekundách

byly zapsány cesty k souborům, zpravidla v místě připojení zkoumaného FS, se kterými měly jednotlivé počítače pracovat při vykonávání testů.

Ve skriptu pojmenovaném „dfsServer“ pak byly implementovány vlastní testy. Skript běžel na všech počítačích clusteru, naslouchal na vybraném portu a plnil požadavky *dfsSingletonu* dokud nebyl explicitně ukončen.

Celý postup při vykonání jednoho testu tedy byl takový:

1. Editace předpřipravených seznamů hostnames počítačů – nastavení účastníků testu. Přitom se rozlišovaly počítače na kterých běželi klienti DFS aktivně vytvářející požadované I/O operace a počítače se spuštěnými pouze datovými resp. meta servery, které test neřídily, ale přesto bylo třeba po jeho ukončení sebrat výsledky sledování hardwaru a vygenerovat z nich grafy (viz dále).

2. Uživatel zadává název testu jako parametr *dfsSingletonu* a spouští jej.

3. *DfsSingleton* odesílá zadání testu spolu s cestami k potřebným souborům všem naslouchajícím procesům *dfsServer* a započíná čekání na příchod výsledků.

4. *DfsServer* přijímá zadání a spouští patřičný test, popřípadě čeká na informaci o ukončení testu, běží-li na neklientské části zkoumaného FS

5. Testy jsou dokončeny. Jednotlivé *dfsServery* spouští shellový skript, který zajistí generování grafů z údajů zaznamenaných nástrojem popsaným v následující sekci.

6. *DfsServer* vrací informace o průběhu a délce testu čekajícímu dfsSingletonu.

7. *DfsSingleton* vypisuje výsledky testů a končí

3.2 Návrh testů

Aby byly systémy porovnatelné i s ostatními, které zpracoval kolega, bylo nutné navrhnout sadu základních testů. Následující kapitoly tedy shrnují postup testování, aby nebylo nutné jej rozebírat u každého systému.

3.2.1 Čtení

Čtení je nejčastější operací se souborovým systémem, proto je i základním testem. Nejprve jsme testovali sekvenční čtení souboru o velikosti 1 GB po úsecích dlouhých 1 MB, přičemž každý klient četl z vlastního souboru. Cílem tohoto testu je určit propustnost datových serverů bez zátěže metadata serveru, jelikož soubor je otevírán pouze jednou. Předpokládá se nejvyšší rychlost právě v tomto testu, protože soubory mohou být umístěny na různých serverech a klienti tak mohou číst každý z vlastního serveru.

Vyvstává zde však problém s poměrně velkou pamětí serverů i klientů. Ta umožňuje ukládání souboru do cache, takže se pak testy výrazně liší od reálného nasazení, kdy se data většinou musí vyhledávat na disku. Využití cache u klienta lze zredukovat jednoduchým způsobem – všichni klienti nejprve zapíší své soubory (což lze rovnou využít jako test zápisu) a poté čtou soubor sousedního klienta, který nemají v paměti. Cache serverů lze obejít buď zápisem a čtením většího množství dat, nebo dvojitým zápisem, kdy každý klient zapíše 2 různé soubory a poté čte první zapsaný.

Dále jsme testovali totéž čtení s jediným rozdílem – všichni klienti četli ze stejného souboru. Toto čtení by mělo být omezeno pouze sítí, pokud datový server přečtená data

uchovává v RAM. V případě prokládání souborů mezi jednotlivé servery by se také mohly jednotlivé části načítat paralelně, čímž by se mohlo dosáhnout vyšší rychlosti než v předchozím testu.

Dalším plánovaným testem bylo čtení souboru o velikosti 1 MB 1000krát za sebou (včetně otevření a zavření). Zde se měla testovat zátěž metadata serverů, ovšem v praxi se spíše uplatnila cache klienta. Výsledky tedy nebyly příliš relevantní, proto jsem přidal test čtení tisíce různých souborů o velikosti 1 MB. Tímto testem se testuje jednak zátěž metadata serveru a jednak náhodné čtení z datových serverů po 1 MB.

3.2.2 Zápis

Současné souborové systémy preferují čtení oproti zápisu, proto by se rozdíly mezi nimi měly projevit zejména při zápisu. Nejprve jsme testovali sekvenční zápis souboru o velikosti 1 GB po úsecích dlouhých 1 MB. Je samozřejmé, že každý klient zapisoval do vlastního souboru. Tímto testem by se měla zjistit propustnost datových serverů opět bez zátěže metadata serveru. Také zde u některých systémů nastal problém s příliš velikou cache na straně serveru, takže zápis byl omezen pouze rychlostí sítě. V tomto případě se však cache dá obejít pouze zápisem většího souboru. Proto jsem přidal další test se zápisem 5 GB od každého klienta.

Dalším plánovaným testem byl zápis 1 MB souboru 1000krát za sebou (včetně otevření a zavření). Ovšem stejně jako u čtení nebyly výsledky příliš relevantní, proto jsem přidal test zápisu tisíce různých souborů o velikosti 1 MB. U tohoto testu se sice opět uplatní cache na straně serverů, ale podle využití paměti a procesoru metadata serveru lze odvodit, jak velkou zátěž by byl schopen zvládat. Zároveň lze porovnat vytížení serveru mezi jednotlivými systémy.

3.2.3 Zápis na konec souboru

Tento druh zápisu je typický pro vytváření datových skladů (viz [3]). Často je potřeba, aby několik klientů zapisovalo konkurentně na konec jednoho souboru, přičemž není bezpodmínečně nutné zachovávat pořadí zápisu. Je nutné pouze zajistit atomičnost jednotlivých zápisů.

Prvním testem tedy byl současný zápis 3 klientů na konec jednoho souboru, kde každý klient zapisoval krátký úsek (1 MB) obsahující stále jeden znak (odlišný od ostatních klientů). Poté se soubor zkontroloval, zda nejsou znaky prokládané. Tímto testem se zkontroluje, zda systém zápisy uskutečňuje opravdu atomicky. V opačném případě (pokud by server prokládal data jednotlivých klientů) by nemělo smysl testovat dále.

Dále jsme testovali současný zápis 4 klientů na konec jednoho souboru, tentokrát však byl test zaměřený na určení výkonu. Každý klient zapisoval 1000krát úsek o délce 1 MB. U tohoto testu už nelze jednoznačně srovnávat všechny systémy, jelikož některé z nich nezajišťují atomičnost zápisu. Jiné systémy (zde například ceph) zase umožňují globálně nastavit neatomickou metodu zápisu (zaměřenou na výkon). Jde tedy spíše o orientační zjištění jakých rychlostí lze dosáhnout, než o porovnání mezi jednotlivými systémy.

3.2.4 Výpadky sítě

Distribuované souborové systémy se nasazují také z důvodu zajištění dostupnosti. Žádný hardware není stoprocentně spolehlivý a je tedy vhodné otestovat, jak se budou systémy chovat v případě výpadku jednoho nebo více serverů.

Výpadek serveru jsme simulovali prostým vypojením síťového kabelu z přepínače. Protože nejčastější zátěží souborového systému je čtení, simulovali jsme výpadek během testu, kdy 4 klienti četli zároveň data ze 4 různých souborů. V provozu byly 4 servery a během testu jsme jeden server odpojili. Další postup se již lišil podle testovaného systému – záleží totiž na tom, zda podporuje replikaci (pak jsme zkoušeli odpojit i další servery).

Výpadek pevného disku jsme nesimulovali, jelikož jsme k disku neměli fyzický přístup a navíc je počítáno s využitím diskového pole s redundancí.

Kapitola 4

Výsledky testování

V následujících kapitolách shrnuji výsledky jednotlivých testů. U každého testu je uvedeno na jaké sestavě byl proveden (tedy počet datových serverů a počet klientů). Poté jsem stručně zhodnotil, čím byl omezen výkon souborového systému. Kde to bylo vhodné, přiložil jsem i graf využití hardware.

4.1 Ceph

Tento systém měl bohužel při testech poměrně nestabilní knihovnu pro připojení klientů, takže se většinu testů bohužel nepodařilo provést. Teprve když „vyšla“ verze 0.2, bylo možné dokončit všechny testy, ale z časových důvodů již nebylo možné detailněji rozebrat statistiky vytížení hardware a určit tak omezující místa systému. Z těchto testů jsou tedy pouze shrnuty výsledky.

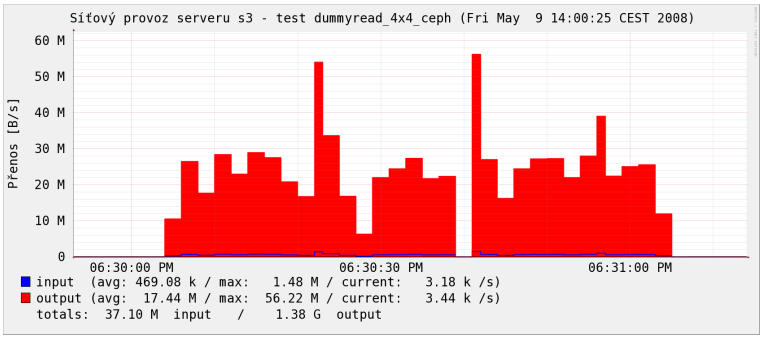
4.1.1 Čtení

Při čtení jednoho klienta ze sestavy 5 serverů (z toho byl 1 metadata server) dosahoval klient rychlosti čtení 45 MB/s. Podle grafů využití sítě četl data ze všech serverů v systému (i když ne rovnoměrně). U jednotlivých serverů bylo využití procesoru a paměti minimální, takže limitujícím faktorem byl patrně disk. Využití sítě u každého serveru bylo nerovnoměrné, stejně jako u následujícího testu.

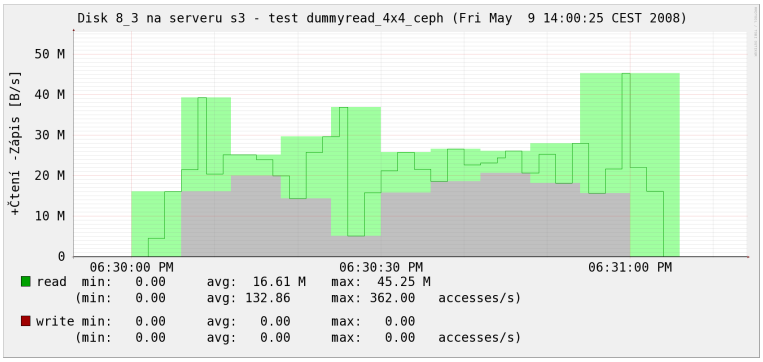
V dalším testu z této sestavy četli zároveň 4 klienti, každý z jiného souboru. Průměrná rychlost každého klienta se pohybovala kolem 17 MB/s. Celkově systém dodával téměř 70 MB/s. Slabým místem systému byly patrně disky jednotlivých serverů, ale jejich zátěž značně kolísala, jak je vidět na grafu 4.2. Pro porovnání je přiložen i graf využití sítě, který přibližně koresponduje s grafem disku, jelikož ceph využívá paměť RAM na cache souborů jen v omezené míře.

4.1.2 Zápis

Při prvním testu zapisoval 1 klient soubor o velikosti 1 GB na stejnou sestavu jako u čtení (4 datové servery a 1 metadata server). Rychlost zápisu dosahovala pouze 8 MB/s. Z grafů využití sítě bylo vidět, že servery si přeposílají data mezi sebou – všechny servery celkově přes síť přijaly 2 GB a odeslaly 1 GB. Soubory jsou tedy evidentně 2krát replikovány pro zajištění redundance a o replikaci se stará cluster. Z grafů však nelze usoudit, zda byla rychlost omezena sítí, nebo pevnými disky. Ani jedno z těchto zařízení totiž nedosahovalo



Obrázek 4.1: Vytížení sítě serveru při čtení.



Obrázek 4.2: Vytížení disku serveru při čtení.

na žádném serveru své maximální propustnosti a podrobnější statistiky nebyly u tohoto testu zaznamenávány (počty paketů a počty přístupů na disk). Jednoznačně lze pouze říci, že procesor a paměť serverů byly využity minimálně.

Při zápisu 4 klientů na totožnou sestavu byla rychlost jednotlivých klientů také 8 MB/s. Celkově systém zapisoval rychlostí 31 MB/s. Ze statistik opět nebylo jednoznačné, co omezovalo rychlost, ale vzhledem k výsledkům předchozího testu bych soudil, že rychlost zápisu je omezována implementací samotného klienta.

Při zápisu 1000 různých souborů jedním klientem se jeho rychlost zápisu zvýšila na 42 MB/s. To je samozřejmě paradox, vzhledem k sekvenčnímu zápisu, ale jak už jsem uvedl výše, rychlost zápisu je patrně omezována implementací. Z tohoto lze soudit, že je klientem omezována rychlost zápisu do jednoho souboru.

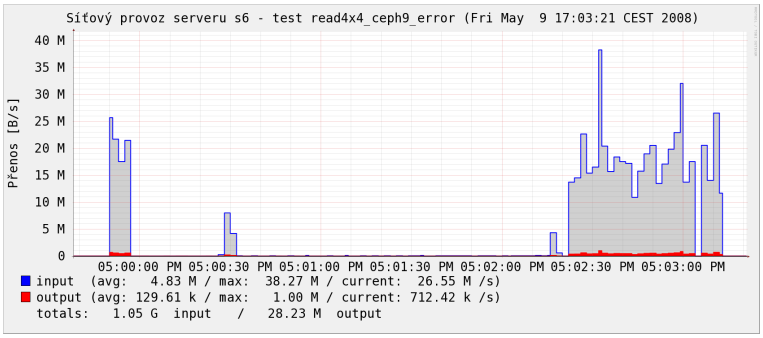
4.1.3 Zápis na konec souboru

Zápis na konec souboru, který byl společný pro všechny klienty, se rychlostně nelišil od obyčejného zápisu do různých souborů. To potvrzuje domněnku, že rychlost zápisu je omezena ze strany klienta.

4.1.4 Výpadky sítě

Při čtení 4 klientů ze 4 serverů jsem jeden ze serverů odpojil. Díky redundanci klienti výpadek prakticky nezaznamenali, pouze se snížila průměrná rychlost čtení každého klienta

na 5 MB/s. Při odpojení druhého serveru, klesla průměrná rychlost pod 4 MB/s. Snížení rychlosti však bylo způsobeno obnovováním replik, jak je vidět z grafu využití sítě klienta 4.3. Po ustálení stavu systému dosahovala rychlost čtení každého klienta 12 MB/s.



Obrázek 4.3: Rychlost čtení klienta při výpadku 1 serveru.

I když jsem oba servery připojil zpátky do sítě, systém je nedetekoval a dál pracoval pouze se dvěma servery. Pokud jsem však odpojil předposlední server, bylo čtení pozastaveno a obnovilo se až po opětovném připojení serveru.

4.1.5 Shrnutí

Tabulka 4.1 shrnuje výsledky jednotlivých testů. Jak bylo uvedeno výše, rychlost zápisu byla pravděpodobně omezena samotnými klienty. V tabulce jsou uvedeny i výsledky naměřené s novou verzí systému, které nebyly rozebrány výše.

Test	1 klient	4 klienti
Zápis 1 GB	7 MB/s	31 MB/s
Zápis do 1 souboru	8 MB/s	31 MB/s
Zápis 1000 x 1MB	43 MB/s	50 MB/s
Čtení 1 GB	45 MB/s	70 MB/s
Čtení z 1 souboru	45 MB/s	99 MB/s
Čtení 1000 x 1MB	35 MB/s	60 MB/s

Tabulka 4.1: Propustnost systému Ceph – 4 datové servery

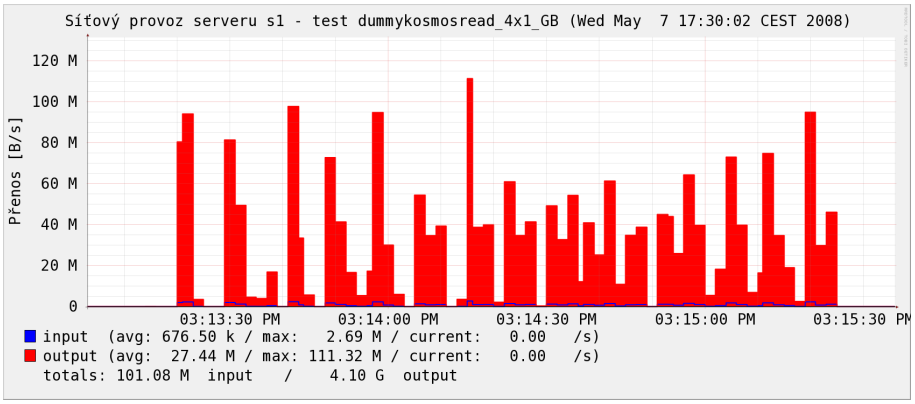
4.2 Kosmos

Tento systém jako jediný neměl podporu pro FUSE, takže bylo nutné použít pro zápis a čtení souborů rozhraní v jazyce C++. Vycházel jsem z testovacích nástrojů poskytovaných přímo autorem souborového systému, které samy o sobě testují zápis, čtení a vytváření adresářů. Tyto nástroje jsem mírně upravil, aby testy odpovídaly testům u ostatních systémů.

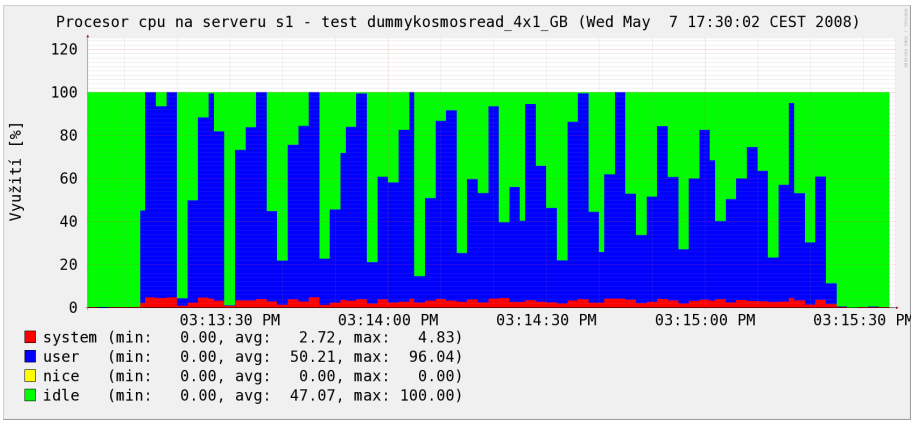
4.2.1 Čtení

Nejprve bylo čtení testováno na sestavě 4 klientů a jednoho serveru (na serveru tedy běžel jak metadata server, tak i datový server). Pokud každý klient četl vlastní soubor, pohybovala

se rychlost čtení jednotlivých klientů pouze okolo 8 MB/s. To odpovídá propustnosti serveru 32 MB/s, což je asi polovina teoretického maxima. Na grafu 4.4 je vidět vytížení sítě serveru. Jsou na něm zřejmé výrazné špičky a výrazné prodlevy mezi nimi. Pokud ke grafu přidáme graf využití procesoru 4.5 je vidět, že prodlevy v síťové komunikaci se kryjí se stoprocentním využitím procesoru. Těmto výkyvům odpovídalo i nárazovité využití disku. Patrně půjde o chybu v implementaci, protože není obvyklé, aby úzkým hrdlem souborového systému byl právě procesor.



Obrázek 4.4: Vytížení sítě serveru při čtení (1 server na 4 klienty)



Obrázek 4.5: Vytížení procesoru serveru při čtení (1 server na 4 klienty)

V dalším testu na stejné konfiguraci četli všichni čtyři klienti stejný soubor. Průměrná rychlost čtení jednotlivých klientů dosahovala 23 MB/s, takže server dodával přes 90 MB/s. To zhruba odpovídá limitu síťové karty, takže omezující byla evidentně síť. Z pevného disku serveru se četlo průměrně 25 MB/s a využití paměti stouplo o 200 MB, takže server patrně v RAM udržoval aktuálně načtené chunks a poskytoval je všem klientům. Zajímavé je, že v tomto případě využití procesoru nepřesáhlo 20%.

Další testy byly prováděny na sestavě 4 klientů a 4 serverů (metadata server běžel na jednom z nich). Pokud měl každý klient vlastní soubor, dosahovali rychlosti čtení v průměru 20 MB/s a celkově systém dodával 78 MB/s. Na jednotlivých serverech byl sice výrazně vytížený procesor, avšak ne na 100%. Síť byla na všech serverech využita nerovnoměrně

a téměř vždy pod 50%. Stejně tak disky byly zatěžovány nerovnoměrně a ani ve špičkách nedodávaly data svou maximální rychlostí, což nasvědčuje neefektivní práci s diskem.

Pokud všichni klienti četli z jednoho společného souboru, rychlost čtení lehce přesahovala 20 MB/s a celková propustnost systému byla 82 MB/s. Z grafů však bylo vidět, že klienti četli všichni z jednoho serveru. Je tedy možné, že jde o chybu testování. Před tímto testem byly testovány výpadky sítě a je možné, že byl čten soubor, který nebyl zreplikován na ostatních serverech.

4.2.2 Zápis

V první testovací sestavě jeden klient zapisoval na 4 servery. Průměrná rychlost zápisu jen mírně překročila 10 MB/s. Z využití sítě bylo vidět, že klient přes síť odeslal 3 GB dat. Patrně tedy musel každou ze tří replik odesílat zvlášť a to výrazně zpomalilo zápis. V pozdějších verzích by si servery měly repliky přeposílat mezi sebou, takže by se rychlost zvýšila na 30 MB/s. I tak je však rychlost hluboko pod limitem sítě, proto jsem hledal další omezení. Na serveru s1, kde byl umístěn metadata server i jeden datový server, byl opět výrazně více využitý procesor (v průměru 20% a ve špičkách 50%, tedy celé jedno jádro) a také disk (viz níže).

V další sestavě 4 klienti zapisovali na 4 servery. Průměrná rychlost zápisu jednoho klienta byla 5,5 MB/s, takže celkově systém zvládal zapisovat 22 MB/s.

V obou případech byl výrazně více využitý procesor a disk byl využit spíše nárazově, což evidentně omezovalo výkon celého systému.

4.2.3 Zápis na konec souboru

Zápis na konec souboru se téměř nelišil od klasického zápisu. Rychlost jednotlivých klientů byla v průměru 5,5 MB/s a celkově systém dodával 22 MB/s. Opět byl na všech serverech výrazně využitý procesor a disk spíše nárazově.

Bohužel z časových důvodů nebyl implementován test atomického zápisu na konec souboru. Na atomickém zápisu si však Google FS zakládá, proto předpokládám, že i v systému Kosmos FS bude implementován.

4.2.4 Výpadky sítě

Asi nejsilnější stránkou tohoto souborového systému by měla být odolnost proti výpadkům jednotlivých serverů. Na „klasické“ testovací sestavě jsem pustil test čtení souboru o velikosti 1 GB a v průběhu odpojil jeden ze serverů. Průměrná rychlost čtení jednotlivých klientů sice klesla na 11 MB/s (oproti běžným 20 MB/s), ale jinak výpadek nebyl znát a všichni klienti dokončili čtení. Proto jsem při dalším testu odpojil rovnou 2 servery. V tomto případě již klienti přestali v průběhu testu číst data. Teprve po opětovném připojení jednoho serveru pokračovali ve čtení, což je v rozporu s nastavenou trojnásobnou replikací (systém by se měl vyrovnat s výpadkem 2 serverů).

4.2.5 Shrnutí

Tabulka 4.2 shrnuje výsledky jednotlivých testů. Uvedena je vždy celková propustnost systému.

Test	1 klient	4 klienti
Zápis 1 GB	10 MB/s	22 MB/s
Zápis do 1 souboru	10 MB/s	23 MB/s
Čtení 1 GB	55 MB/s	78 MB/s
Čtení z 1 souboru	55 MB/s	82 MB/s

Tabulka 4.2: Propustnost systému Kosmos – 4 datové servery

4.3 Lustre

Tento systém se sice testoval díky svojí stabilitě nejsnáze, ale vyžadoval formátování diskových oddílů vlastním souborovým systémem. Také nepočítá s odebíráním jednotlivých datových serverů, takže nebylo možné za běhu měnit počet serverů v sestavě.

4.3.1 Čtení

Při prvním testu četl jeden klient ze 4 serverů a dosáhl při tom rychlosti čtení 48 MB/s. Ze statistik sítě bylo poznat, že četl data z jediného serveru, takže limitující byla rychlost pevného disku tohoto serveru.

Pak jsem čtení testoval na sestavě 4 klientů a 4 serverů (1 metadata server a 3 datové servery), přičemž každý klient četl vlastní soubor. U klientů byly značné rozdíly v rychlosti čtení – první dva klienti dosáhli rychlosti 48 a 42 MB/s, další dva klienti četli rychlostí 24 MB/s. Celkově tedy systém dodával data rychlostí 138 MB/s. Důvod, proč byla rychlost u dvou klientů poloviční, je zřejmý z využití sítě serveru s4. Tito dva klienti totiž četli soubory, které byly umístěny na stejném datovém serveru (tj. s4). Protože při tomto testu nebylo zapnuté prokládání souborů, museli si oba klienti rozdělit rychlost disku jednoho serveru.

Na stejné sestavě jsem testoval i čtení 4 klientů z jednoho souboru. Zde rychlost čtení jednotlivých klientů jen mírně přesahovala 10 MB/s. Všichni klienti četli samozřejmě z jednoho serveru a podle využití disku a paměti lze usoudit, že server neukládal žádná data do paměti RAM. Klienti se tedy museli dělit o rychlost disku. Využití cache v paměti RAM by sice rychlost značně navýšilo, ale v praxi se stává zřídka, aby klienti četli některý soubor zároveň. Pozitivní je, že i v případě využití disku více klienty zůstala jeho propustnost celkem vysoká (přes 40 MB/s).

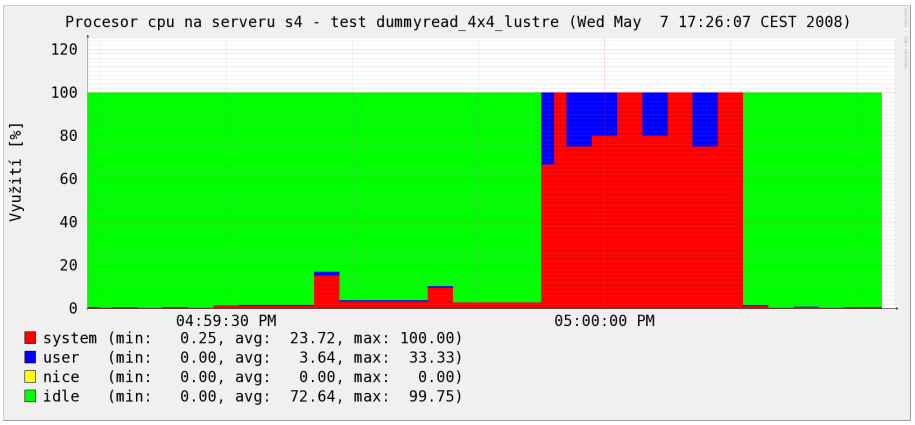
Čtení s rozkládáním souborů

Jak je však vidět, bez prokládání souborů rychlosti značně kolísají a jeden klient nemá možnost využít celý cluster, pokud nečte paralelně více souborů (ale i poté se může stát, že všechny soubory budou umístěné na 1 serveru). Proto jsem systém rekonfiguroval, aby soubory vždy rozkládal mezi 3 servery. Zároveň jsem přidal jeden datový server, aby bylo možné porovnávat s ostatními systémy (i když zůstává drobná výhoda pro lustre – metadata

server je umístěn na samostatném stroji). Následující testy jsou tedy na sestavě 5 serverů (z toho 1 je pouze metadata server) a s nastaveným rozkládáním souborů na 3 části.

Při čtení jednoho klienta z této sestavy (klient četl 1GB dat) dosahoval klient rychlosti čtení 100 MB/s. Rychlost byla tedy evidentně omezena pouze síťovou kartou klienta a klient plně využil celý systém[1].

Dále jsem testoval čtení čtyř klientů, kdy každý klient četl vlastní soubor. Rychlost jednotlivých klientů byla v průměru 38 MB/s. To je sice o něco méně než maximum v testu bez rozkládání, ale na druhou stranu touto rychlostí četli všichni klienti, takže celková propustnost systému přesáhla 150 MB/s. Je těžké soudit, čím byla rychlost čtení omezena, ale patrně byly úzkým místem disky jednotlivých serverů. Každý z nich dodával stabilně necelých 40 MB/s, což je stejně jako v předchozích testech a pravděpodobně tedy i maximální rychlost čtení. Zarážející jsou však výrazné výkyvy ve využití procesoru, například na serveru s4 (viz graf 4.6). Buď jde o chybu při měření, nebo jádro systému využívá procesor takřka na 100%.



Obrázek 4.6: Vytížení procesoru serveru při čtení (4 servery na 4 klienty)

Pokud klienti četli jeden společný soubor, průměrná rychlost čtení jednoho klienta se pohybovala nad 27 MB/s. Celkově tedy systém dodával 110 MB/s, což zhruba odpovídá tomu, že klienti četli pouze ze 3 datových serverů. Opět však byl výrazně více využitý procesor.

V posledním testu každý klient četl tisíc různých souborů, každý o délce 1 MB. V tomto případě se průměrná rychlost čtení každého klienta pohybovala těsně pod 15 MB/s. Paradoxně úzkým místem systému nebyl metadata server (alespoň podle využití sítě a procesoru – obojí pod 2%). Pevný disk datových serverů také patrně nebyl omezující, jelikož průměrný počet přístupů k disku na žádném serveru nepřesáhl 50 přístupů/s. Disky jednotlivých serverů, přesto dodávaly pouze 15 MB/s.

4.3.2 Zápis

První test byl na sestavě čtyř klientů a čtyř serverů, bez nastaveného rozkládání souborů. Průměrné rychlosti zápisu klientů byly 48 MB/s, 38 MB/s, 21 MB/s a 21 MB/s. Nižší rychlost zápisu u posledních 2 klientů byla opět způsobená tím, že zapisovali oba na jeden datový server.

[1] Při faktoru prokládání 2 totiž byla rychlost pouze 80 MB/s – tedy stále omezená disky serverů

Všechny další testy už jsem prováděl na sestavě 4 klientů a 5 serverů, s nastaveným rozkládáním souborů mezi 3 servery.

Nejprve jsem testoval zápis jednoho klienta. Dle očekávání byla rychlost zápisu (téměř 100 MB/s) omezena pouze sítí. V dalším testu na systém zapisovali 4 klienti. Rychlost zápisu byla takřka stejná jako rychlost čtení, u každého klienta průměrně 37 MB/s. Celkově tedy systém zapisoval téměř 150 MB/s. Limitující byly pravděpodobně pevné disky jednotlivých serverů, jelikož každý server musel zapisovat téměř 40 MB/s a Lustre nevyužíval paměť RAM na cache.

Při posledním testu každý klient zapisoval 1000 různých souborů. Nejprve se zapisovaly soubory velikosti 1 MB, takže rychlost zápisu závisela především na datových serverech. V tomto případě trval test jednotlivým klientům v průměru 108 sekund, což odpovídá rychlosti zápisu přibližně 9 MB/s. Zátěž metadata serveru byla ještě nižší než v případě čtení, takže téměř nulová. Limitující tedy byly opět datové servery. Samotné vytvoření tisíce souborů délky 1 B trvalo přibližně 3 sekundy, což je málo na vyvozování dalších závěrů (vzhledem k rozlišení statistik 1 s).

4.3.3 Zápis na konec souboru

Tyto testy jsem všechny prováděl na sestavě čtyř datových serverů a jednoho metadata serveru, s nastaveným rozkládáním souborů mezi 3 servery. Nejprve jsem zkoušel původně navrhnutý test – zápis tří klientů na konec jednoho souboru, kdy každý klient zapisoval úsek délky 1 MB. Výsledný soubor obsahoval zápisy jednotlivých klientů za sebou. Sice nebyly uspořádané podle pořadí zápisu, ale podstatné je, že nebyly prokládané.

Dále jsem zkoušel totožný test, pouze s delšími soubory (1 GB). Z grafů využití sítě bylo poznat, že klienti zapisovali postupně po sobě a každý z nich rychlostí 100 MB/s. Rychlost zápisu tedy byla omezena síťovou kartou klientů a zamčením souboru (vždy pro jednoho klienta).

4.3.4 Výpadky sítě

Systém Lustre obecně nepočítá s častějšími a dlouhodobějšími výpadky HW, což se projevilo i při testování. Jakmile jsem při čtení odpojil jeden server, přestali číst všichni klienti, kteří četli soubor z tohoto serveru. U ostatních klientů se však výpadek vůbec neprojevil. Když bylo nastavené rozkládání souborů, výpadek se dotknul všech klientů, protože soubory četli sekvenčně. Podle dokumentace by měly být čitelné všechny zbylé části souboru a náhodným přístupem by tedy bylo možné přečíst alespoň část souboru. Pokud se v průběhu čtení odpojil metadata server, u klientů se to neprojevilo vůbec. To odpovídá architektuře systému, kdy se z metadata serveru určí pouze umístění souboru na základě jeho názvu. Pochopitelně však nebylo možné nově otevírat soubory ani procházet adresářovou strukturu.

Ve všech předcházejících případech se zhruba do 10 sekund od opětovného připojení serveru obnovila veškerá činnost systému, včetně přerušeného čtení.

4.3.5 Shrnutí

Tabulky 4.3 a 4.4 shrnují výsledky jednotlivých testů. Uvedena je vždy celková propustnost systému. Tabulka 4.3 obsahuje výsledky dosažené na systému složeném ze 3 datových serverů a jednoho metadata serveru, na kterém nebylo nastaveno rozkládání souborů mezi více serverů. Následující tabulka (4.4) obsahuje výsledky dosažené na sestavě 4 datových serverů a jednoho metadata serveru, s nastaveným rozkládáním souborů mezi 3 servery.

Test	**1 klient**	**4 klienti**
Zápis 1 GB	46 MB/s	152 MB/s
Zápis do 1 souboru	43 MB/s	41 MB/s
Čtení 1 GB	48 MB/s	138 MB/s
Čtení z 1 souboru	48 MB/s	41 MB/s

Tabulka 4.3: Propustnost systému bez rozkládání souborů.

Test	**1 klient**	**4 klienti**
Zápis 1 GB	100 MB/s	148 MB/s
Zápis do 1 souboru	100 MB/s	38 MB/s
Zápis 1000 x 1MB	8 MB/s	38 MB/s
Čtení 1 GB	100 MB/s	152 MB/s
Čtení z 1 souboru	100 MB/s	110 MB/s
Čtení 1000 x1MB	22 MB/s	59 MB/s

Tabulka 4.4: Propustnost systému s rozkládáním souborů.

Kapitola 5

Shrnutí

V následující tabulce (5.1) jsem shrnul vlastnosti testovaných souborových systémů. Tabulka je uvedena zejména kvůli možnosti porovnání s ostatními systémy, které zpracovával kolega (viz [4]).

Vlastnost	**Ceph**	**Kosmos FS**	**Lustre**
Licence	GNU LGPL	Apache 2.0	GNU GPL
Stabilní	Ne	Ne	Ano
Metadata server	Ano	Ano	Ano
Replikace	Ano	Ano	Ne
Rozkládání	Ano	Ano	Volitelné
Připojení do VFS	FUSE, modul jádra	FUSE (nestabilní)	modul jádra
Propojení	TCP/IP	TCP/IP	TCP/IP a jiné
Výpadek	Omezena rychlost	Omezena rychlost	Pozastaveno
Append	Souvislý	Souvislý	Souvislý
Zápis 1 GB	31 MB/s	22 MB/s	148 MB/s
Zápis do 1 souboru	31 MB/s	23 MB/s	38 MB/s
Zápis 1000 x 1 MB	50 MB/s	—	38 MB/s
Čtení 1 GB	70 MB/s	78 MB/s	152 MB/s
Čtení 1 souboru	99 MB/s	82 MB/s	110 MB/s
Čtení 1000 x 1 MB	60 MB/s	—	59 MB/s

Tabulka 5.1: Srovnání vlastností souborových systémů.

Významy jednotlivých položek jsou následující:

- Licence – pod jakou licencí je systém vydán.
- Stabilita – zda je podle autorů systém vhodný k nasazení v praxi.
- Metadata server – zda je pro metadata vyhrazen zvláštní server (případně servery)
- Replikace – zda je zajištěna bezpečnost dat replikací.
- Rozkládání – zda jsou jednotlivé soubory rozkládány mezi více serverů.
- Připojení do VFS – možnosti pro připojení do souborového systému klienta.

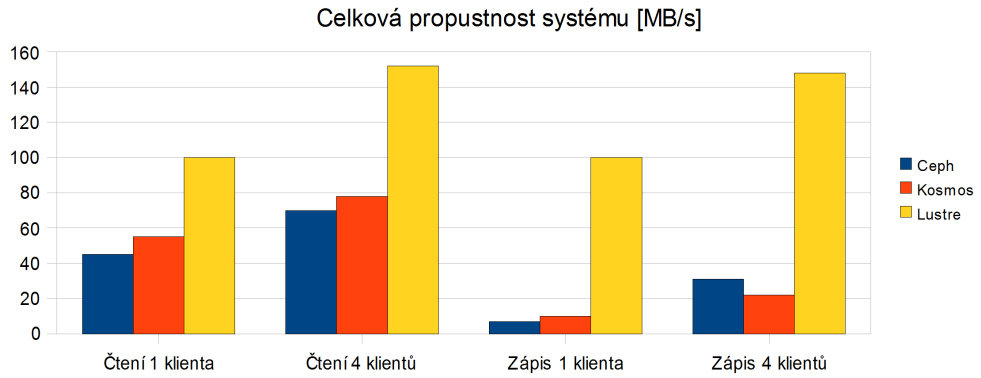
- Propojení – rozhraní, kterými je možné propojovat uzly systému.
- Výpadek – zda bylo možné pokračovat ve čtení při výpadku datového serveru.
- Append – zda byl zápis jednotlivých klientů na konec souboru souvislý (atomický).
- Rychlosti – na sestavě 4 klientů a 4 serverů. Je uvedena celková propustnost systému.

Kapitola 6

Závěr

Cílem práce bylo porovnat vybrané souborové systémy, otestovat je na referenčním hardware a zhodnotit možnost nasazení těchto systémů na fakultě.

Na tomto tématu jsem pracoval zároveň s Vítězslavem Humpou, který ve své práci [4] zpracoval další souborové systémy. Systémy jsem otestoval sadou testů na zapůjčeném hardware a podle záznamů o využití hardware jsem vyhodnotil, čím byl výkon omezen. Velmi stručné shrnutí výsledků testů je v grafu 6.1.



Obrázek 6.1: Stručné shrnutí výsledků.

Prvním porovnávaným systémem byl Ceph. Výhodou tohoto systému je, že je určen pro obecné použití a není zaměřen na žádný specifický typ zátěže. Další výhodou je možnost replikace dat, čímž je zajištěna odolnost proti výpadku serveru. Nevýhodou může být nižší výkon při specifické zátěži a nestabilita systému, jelikož je zatím ve vývoji. Systém bych tedy hodnotil jako perspektivní do budoucna, ale zatím nevhodný pro nasazení v praxi.

Dalším systémem byl Kosmos FS, který je nyní v počátcích vývoje. Je sice poměrně stabilní a díky replikaci dat i spolehlivý, ale chybí mu podpora přístupových práv a FUSE. Nedosahuje ani výrazně vyšších výkonů, takže bych zatím nedoporučoval jeho nasazení.

Posledním srovnávaným systémem byl Lustre. Jeho hlavní výhodou je dlouhodobé nasazení v produkčním prostředí a z toho vyplývající stabilita a výkon. Nevýhodou je nutná úprava jádra na straně serveru a jeho zaměření na velké soubory. Při práci s malými soubory je výkon znatelně nižší. Z porovnávaných systémů dosahoval nejvyšších výkonů a je tedy nejvhodnějším kandidátem pro nasazení na fakultě.

Protože jsem zpracovával více souborových systémů, zabýval jsem se jejich konfigurací jen povrchně. V případě nasazení některého souborového systému by bylo vhodné otestovat i jeho chování při různých konfiguracích, například s různými faktory rozkládání souborů. Dále by bylo vhodné tyto systémy otestovat v budoucnu ještě jednou, až budou ve stabilnější verzi.

Literatura

[1] Ceph wiki. Dostupné na URL http://ceph.newdream.net/wiki/Main_Page (duben 2007).

[2] Kim Cupps. *Blue Gene/L System Software Update*. 2005. UCRL-PRES-210094.

[3] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In Proceedings of the 19th ACM Symposium on Operating Systems Principles. Lake George, NY, 2003.

[4] Vítězslav Humpa. *Porovnání vlastností distribuovaných souborových systémů*. FIT VUT v Brně, 2008. Bakalářská práce.

[5] Sun Microsystems. *Lustre file system*. 2007. Dostupné na URL http://www.sun.com/servers/hpc/docs/lustrefilesystem_wp.pdf (duben 2008).

[6] D. Roselli, J. Lorch, and T. Anderson. A comparison of file system workloads. In Proceedings of the 2000 USENIX Annual Technical Conference, pages 41–54. San Diego, CA, 2000.

[7] Sage Weil. Posix file system test suite. Dostupné na URL http://ceph.newdream.net/blog/2008/04/18/posix-file-system-test-suite/ (duben 2008).

[8] Sage Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Carlos Maltzahn. Ceph: A scalable, high-performance distributed file system. Proceedings of the 7th Conference on Operating Systems Design and Implementation. Springer, 2006.

[9] Sage A. Weil, Andrew W. Leung, Scott A. Brandt, and Carlos Maltzahn. *RADOS: A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters*. University of California, Santa Cruz, 2005.